Flicker: An Execution Infrastructure for TCB Minimization

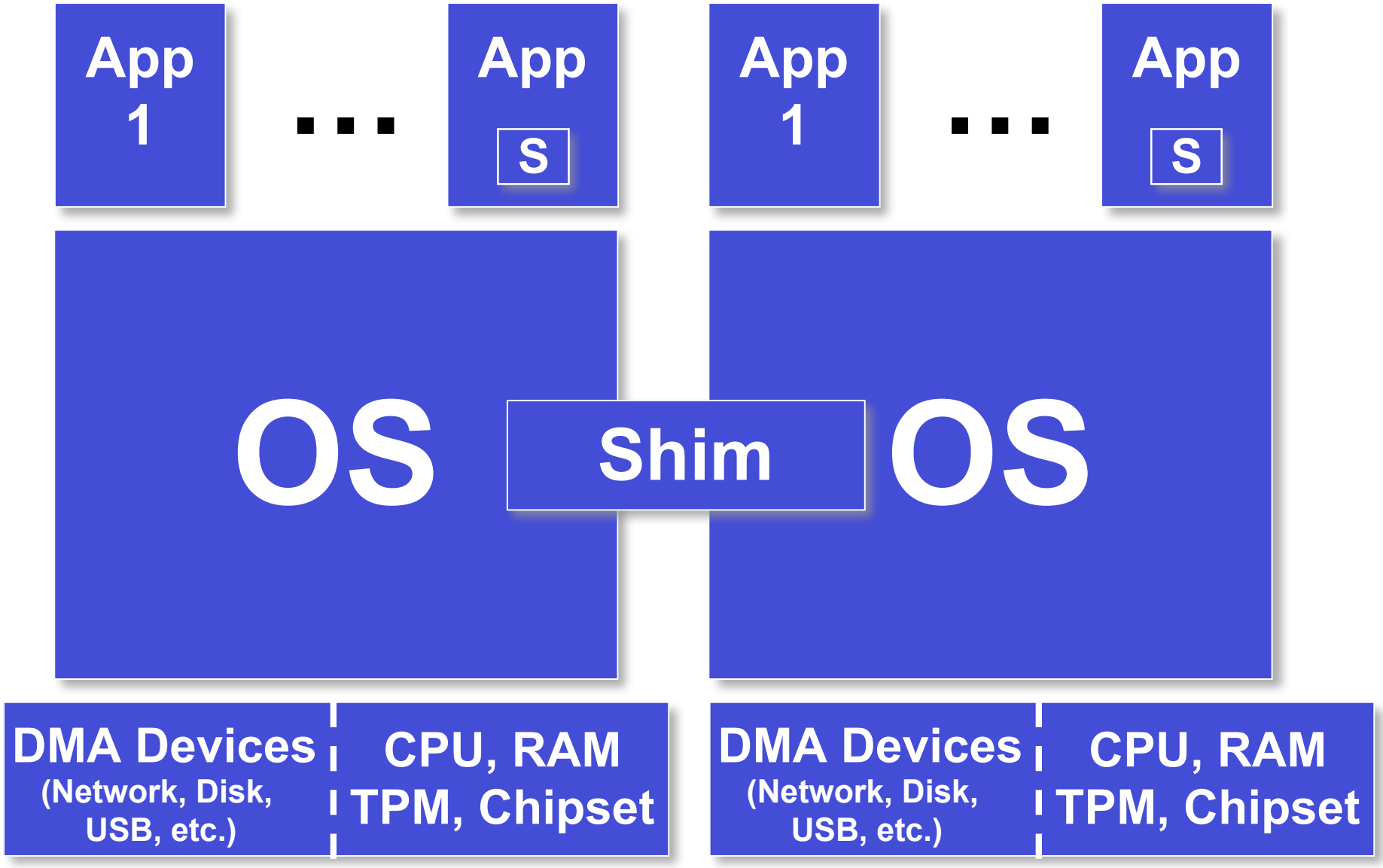
Jonathan McCune[1], Bryan Parno[1], Adrian Perrig[1], Michael Reiter[2], and Hiroshi Isozaki[1,3]

[1] *Carnegie Mellon University*
[2] *University of North Carolina*
[3] *Toshiba Corporation*

April 3, 2008

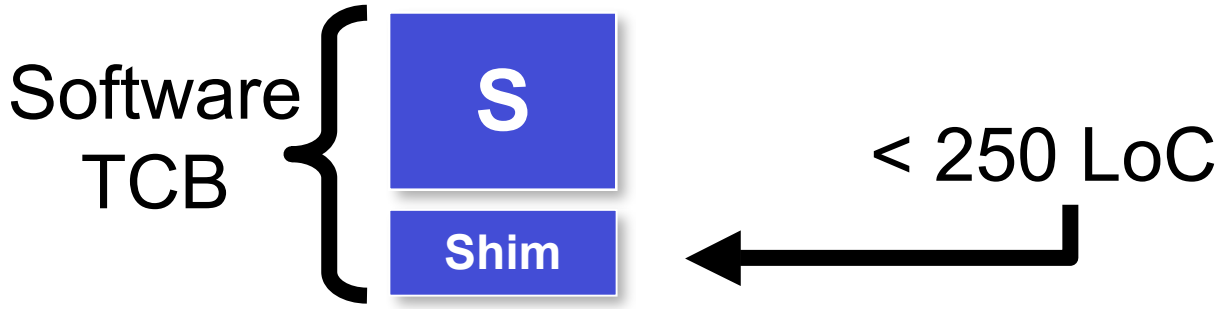
Berkeley UNIVERSITY OF CALIFORNIA
Carnegie Mellon
Cornell University
MILLS COLLEGE
San José State UNIVERSITY
SMITH COLLEGE
STANFORD UNIVERSITY
VANDERBILT UNIVERSITY

Trusted Computing Base (TCB)



Flicker's Properties

- Isolate security-sensitive code execution from all other code and devices
- Attest to security-sensitive code and its arguments and nothing else
- Convince a remote party that security-sensitive code was protected
- Add < 250 LoC to the software TCB



Outline

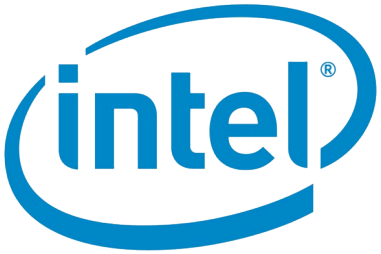
- Introduction
- Background
  - Trusted Platform Module (TPM)
  - Late Launch
- Flicker Architecture and Extensions
- Flicker Applications
- Performance Evaluation
- Related Work and Conclusions

TPM Background

- The Trusted Platform Module (TPM) is a dedicated security chip
- Can provide an *attestation* to remote parties
  - Platform Configuration Registers (PCRs) summarize the computer's software state
    - PCR_Extend(N, V): $PCR_N = SHA\text{-}1(PCR_N \mid V)$
  - TPM provides a signature over PCR values
  - A subset of *dynamic* PCRs can be reset without a reboot

Late Launch Background

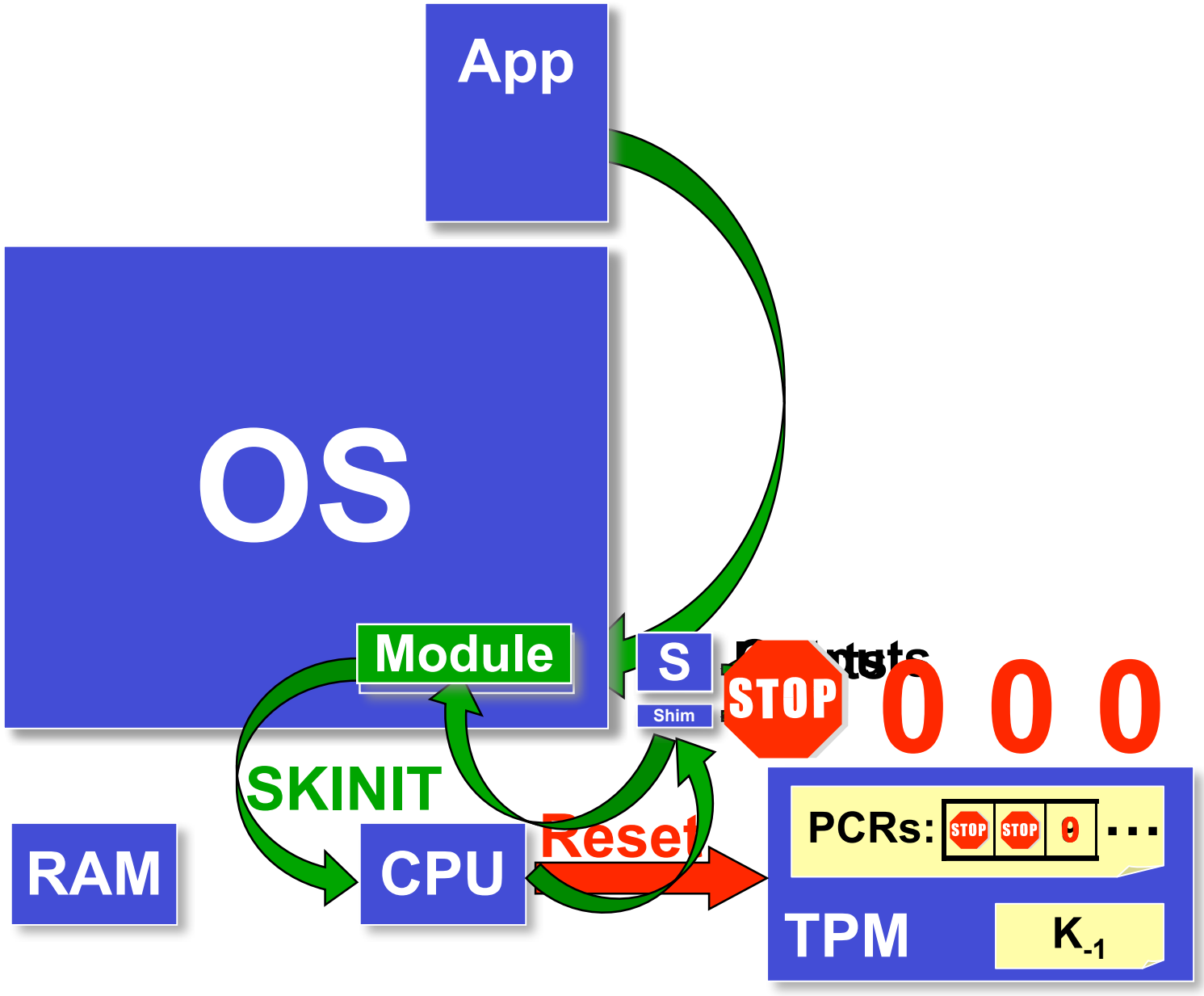
- Supported by new commodity CPUs
  - SVM for AMD
  - TXT (formerly LaGrande) for Intel
- Designed to launch a VMM without a reboot
  - Hardware-based protections ensure launch integrity
- New CPU instruction (SKINIT/SENTER) accepts a memory region as input and atomically:
  - Resets dynamic PCRs
  - Disables interrupts
  - Extends a measurement of the region into PCR 17
  - Begins executing at the start of the memory region



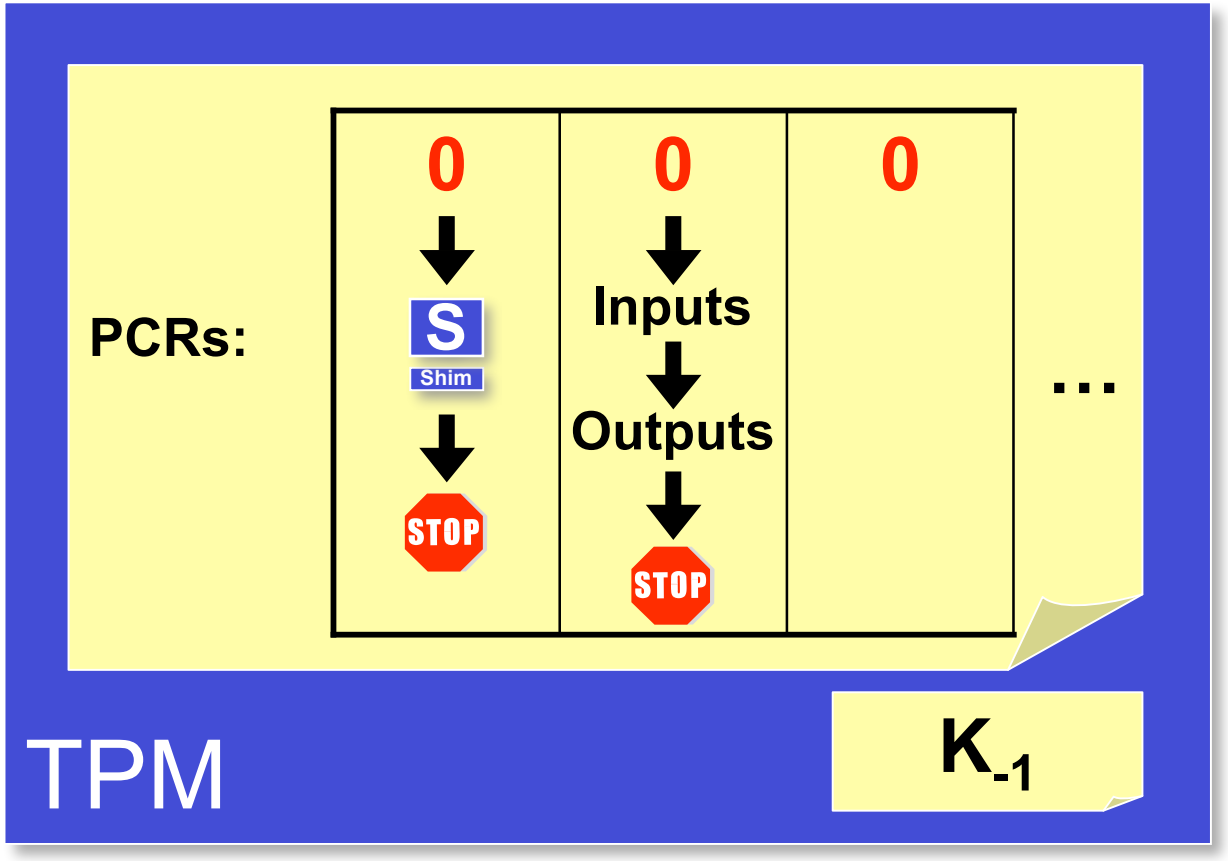
Architecture Overview

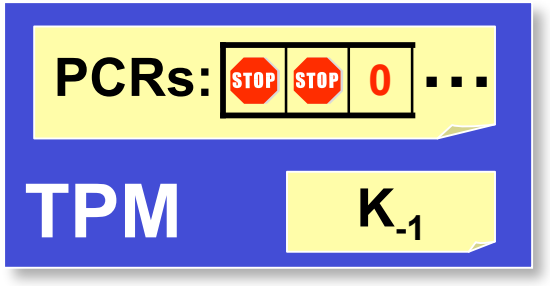
- Core technique
  - Pause current execution environment (untrusted OS)
  - Execute security-sensitive code with hardware-enforced isolation
  - Resume previous execution
- Extensions
  - Attest **only** to code execution and protection
  - Preserve state securely across invocations
  - Establish secure communication with remote parties

Execution Flow

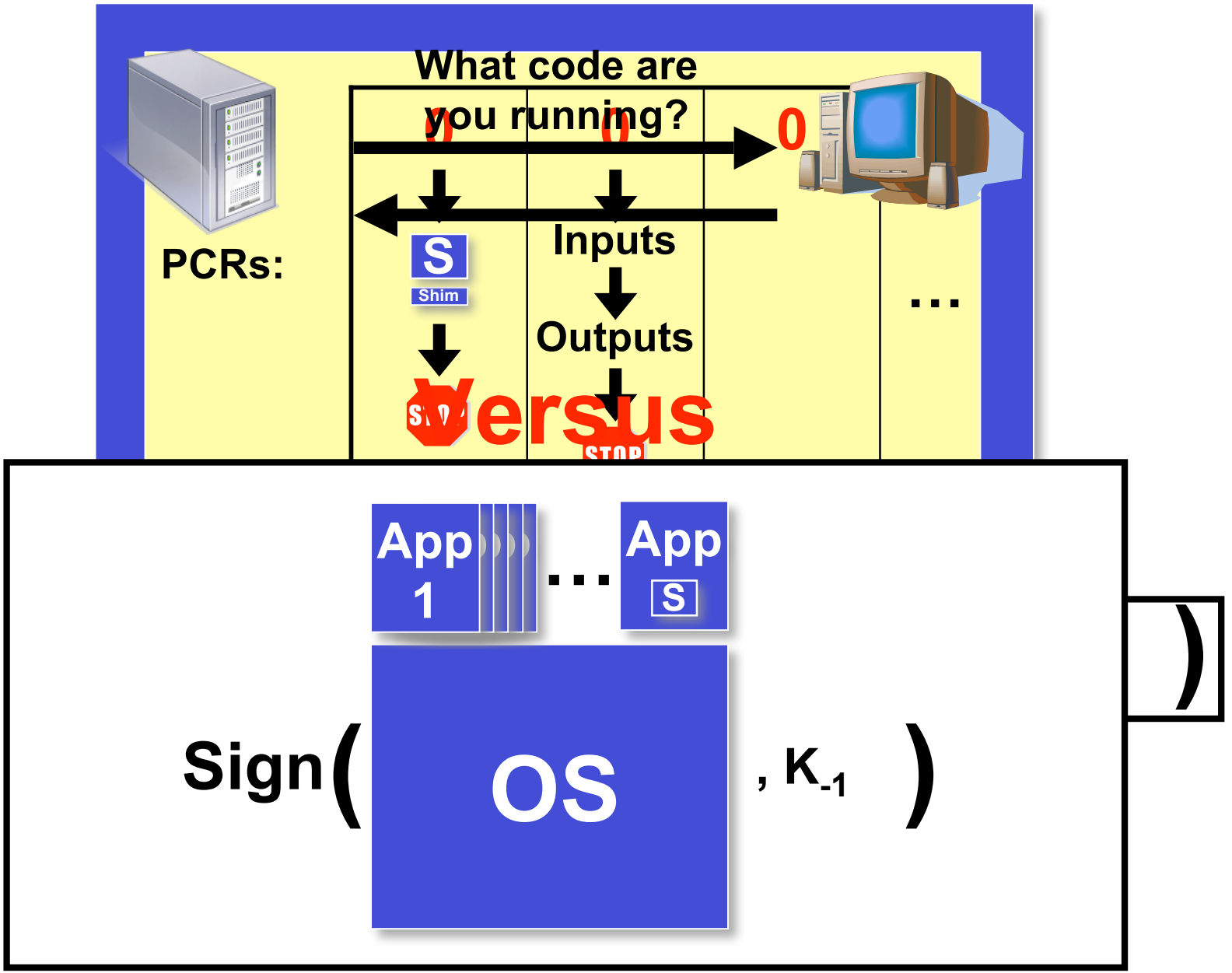


Attestation



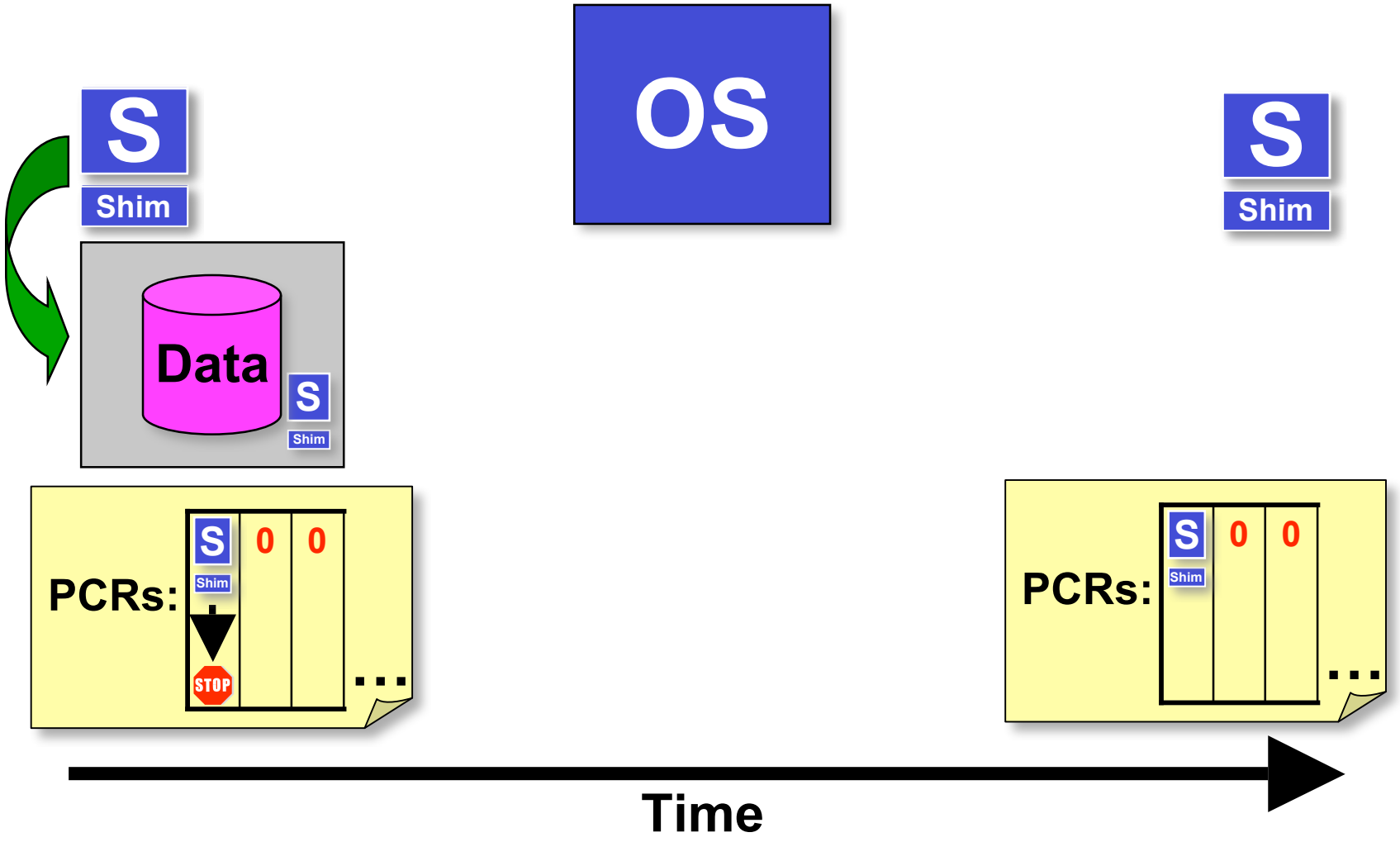


Attestation



Context Switch with Sealed Storage

- **Seal** data under combination of code, inputs, outputs
- Data unavailable to other code



Outline

- Introduction
- Background
- Flicker Architecture and Extensions
- Flicker Applications
  - Developer's Perspective
  - Example Applications
- Performance Evaluation
- Related Work and Conclusions

Developing With Flicker

- Sensitive code linked against the Flicker library
- Customized linker script lays out binary
- Application interacts with Flicker via a Flicker kernel module

```
#include "flicker.h"
const char* msg = "Hel
void flicker_main(void *      uts) {
      for(int i=0;i<13;i++)
          OUTPUT[i] = msg[i];
}
```

Made available at: /proc/flicker/output

Default Functionality

- Shim can execute arbitrary x86 code but provides very limited functionality
- Fortunately, many security-sensitive functions do not require much
  - E.g., key generation, encryption/decryption, FFT
- Functionality can be added to support a particular security-sensitive operation
- We have partially automated the extraction of support code for security-sensitive code

Existing Flicker Modules

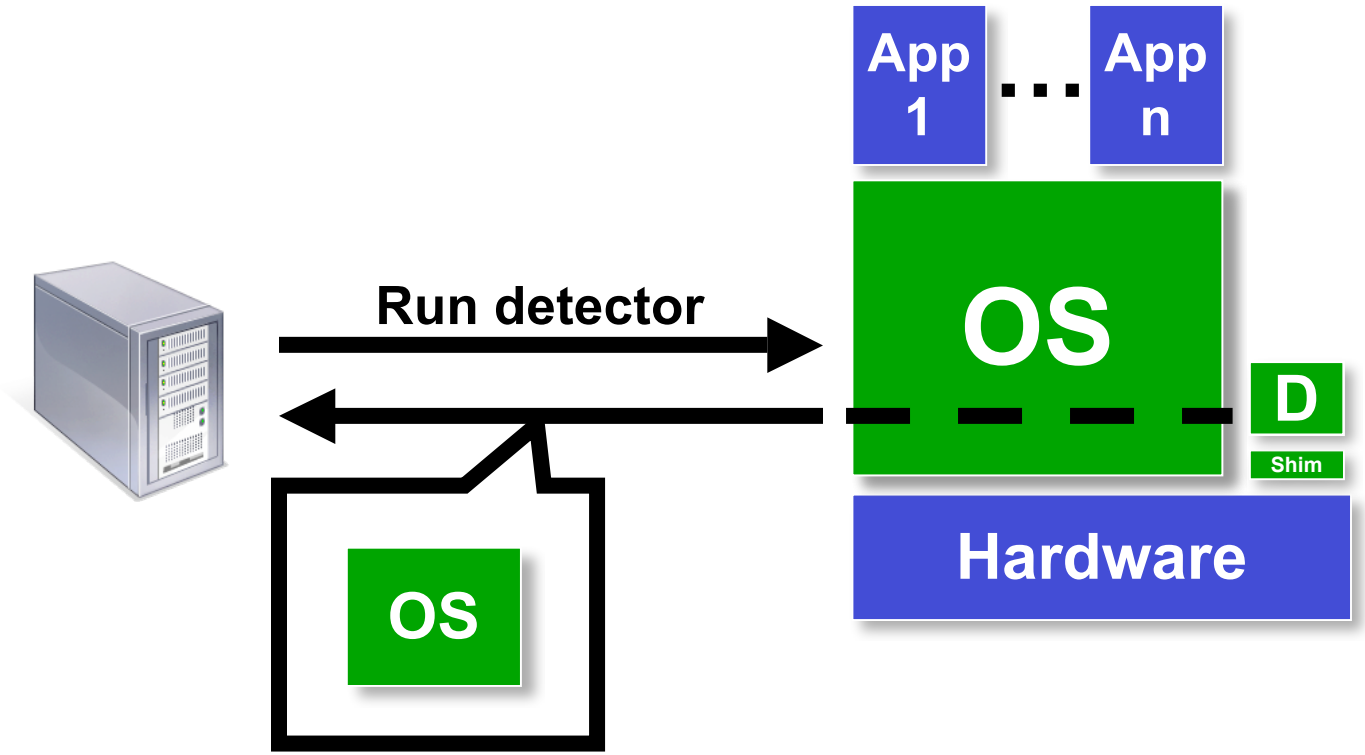
- **OS Protection** Memory protection, ring 3 execution
- **Crypto** Crypto ops (RSA, SHA-1, etc.)
- **Memory Alloc.** Malloc/free/realloc
- **Secure Channel** Secure remote communication
- **TPM Driver** Communicate with TPM
- **TPM Utilities** Perform TPM ops

Outline

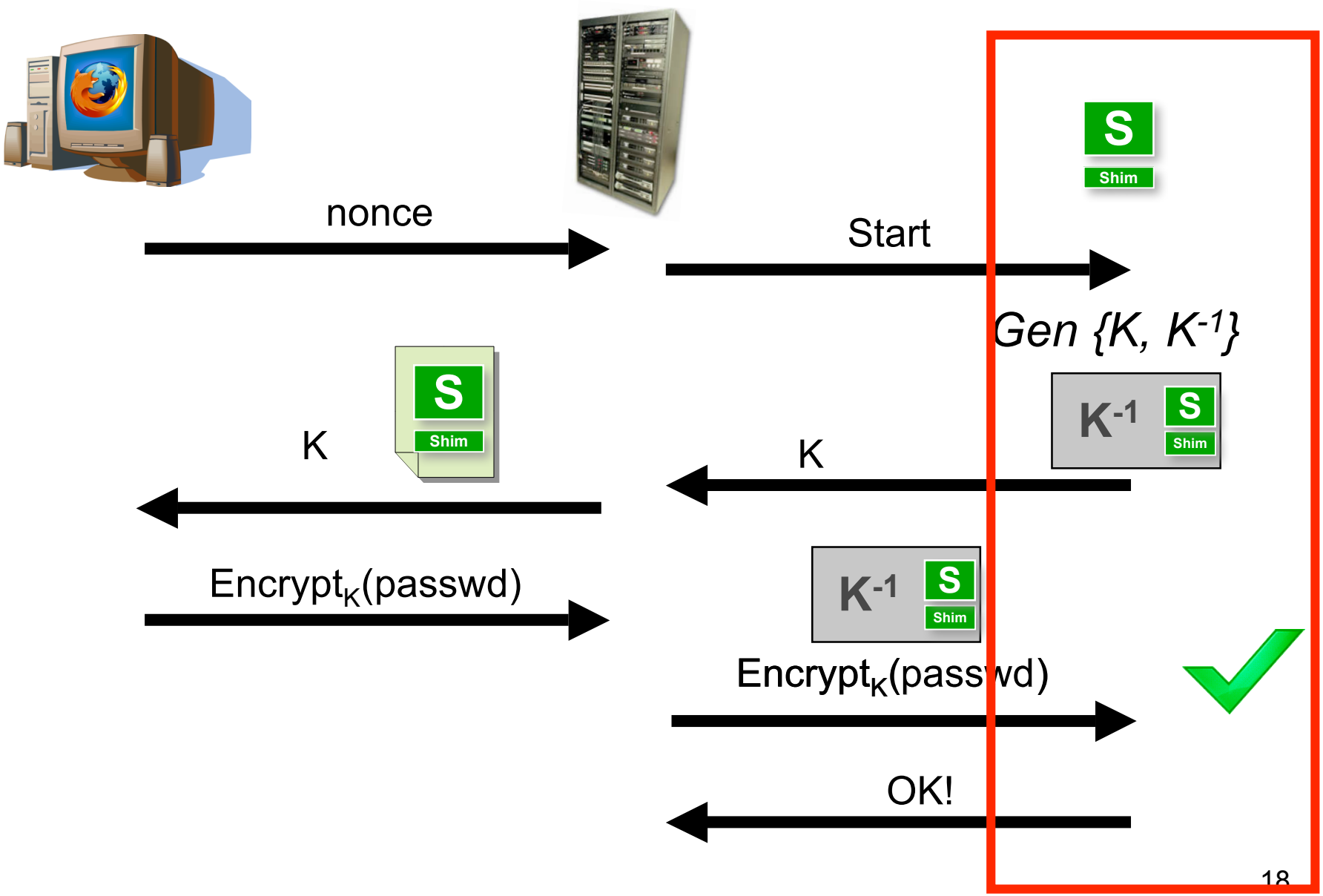
- Introduction
- Background
- Flicker Architecture and Extensions
- Flicker Applications
  - Developer's Perspective
  - Example Applications
- Performance Evaluation
- Related Work and Conclusions

Application: Rootkit Detector

- Administrator can check the integrity of remote hosts
  - E.g., only allow uncompromised laptops to connect to the corporate VPN



Application: SSH Passwords



Other Applications Implemented

- Enhanced Certificate Authority (CA)
  - Private signing key isolated from entire system
- Verifiable distributed computing
  - Verifiably perform a computational task on a remote computer
  - Ex: SETI@Home, Folding@Home, distcc

Outline

- Introduction
- Background
- Flicker Architecture and Extensions
- Flicker Applications
- Performance Evaluation
- Related Work and Conclusions

Generic Context-Switch Overhead

- Each Flicker context switch requires:
  - SKINIT
  - TPM-based protection of application state

Results

SKINIT	14 ms
Unseal application state	905 ms
Reseal application state	20 ms
Total	**939 ms**

Rootkit Detection Performance

SKINIT	14 ms
Hash of Kernel	22 ms
PCR Extend Result	1 ms
TPM Quote	973 ms
Total	**1023 m**

37 ms Disruption

Non-Disruptive

Running detector every 30 seconds has negligible impact on system throughput

SSH Performance

- Setup time (217 ms) dominated by key generation (185 ms)
- Password verification (937 ms) dominated by TPM Unseal (905 ms)

Adds < 2 seconds of delay to client login

Optimizing Flicker's Performance

- Non-volatile storage
  - Access control based on PCRs
  - Read in 20ms, Write in 200 ms
  - Store a symmetric key for "sealing" and "unsealing" state

Reduces context-switch overhead by an order of magnitude

Hardware Performance Improvements

[ASPLOS 2008]

- Late launch cost only incurred when Flicker session launches
- TPM (Un)Seal only used for long-term storage
- Multicore systems remain interactive
- Context switch overheads (common case) resemble VM switches today (~0.5 µs)

Ongoing Work

- Creating a trusted path to the user
- Porting implementation to Intel
- Improving automatic privilege separation

Related Work

- Secure coprocessors
  - Dyad [Yee 1994], IBM 4758 [JiSmiMi 2001]
- System-wide attestation
  - Secure Boot [ArFaSm 1997], IMA [SaZhJaDo 2004], Enforcer [MaSmWiStBa 2004]
- VMM-based isolation
  - BIND [ShPeDo2005], AppCores [SiPuHaHe 2006], Proxos [TaLiLi 2006]
- "Traditional" uses of late launch
  - Trustworthy Kiosks [GaCáBeSaDoZh 2006], OSLO [Kauer 2007],

Conclusions

- Flicker greatly reduces an application's TCB
- Isolate security-sensitive code execution
- Provide fine-grained attestations
- Allow application writers to focus on the security of their own code